Airfoil Design Using Navier-Stokes Equations and Hybrid Evolutionary Optimization Techniques

Domenico Quagliarella[‡], **Domenic D'Ambrosio**[⋆], **Angelo Iollo**[⋆]
[‡] Centro Italiano Ricerche Aerospaziali
Via Maiorise, 81043 Capua, Italy
[⋆]Dipartimento di Ingegneria Aeronautica e Spaziale, Politecnico di Torino
C.so Duca degli Abruzzi 24, 10129 Torino, Italy
d.quagliarella@cira.it, domenic@athena.polito.it, iollo@polito.it

1 Introduction

An application of hybrid optimization techniques to airfoil design with Navier-Stokes equations is here presented and discussed. The greatest obstacle in the use of Navier-Stokes equations with evolutionary optimization procedures is the huge computational effort required. To overcome this limit, several approaches have been suggested, that range form parallel genetic algorithms [1, 2] to approximated fitness evaluation [3, 4] to hybrid techniques [5].

These three approaches can be profitably combined and an example will be here given related to transonic airfoil design.

The lecture is organized as follows: an outline of hybrid optimization techniques will be given then the parallel hybrid genetic algorithm will be presented. Thereafter an application example will be reported, and finally some ongoing developments about the hybrid approach based on gradient computation through the adjoint to Navier-Stokes equations will be described.

2 Hybrid genetic algorithms

Hybridization has been one of the first techniques adopted for improving genetic algorithm performance, while keeping the desirable flexibility features of genetic algorithms [6].

In the optimization context, hybridization can be defined as the mixing of two or more search techniques with, possibly, complementary features.

The genetic algorithm itself, even in its simplest implementation, can be considered as a hybrid optimization technique, where selection, mutation and crossover cooperate in the same optimization process.

Figures 1 and 2 show how the genetic algorithm can be extended through the addition of a hill climbing operator. In the first scheme the hill climber receives in input the whole population belonging to the current generation, while in the second one an intermediate selection process chooses a subset of elements that will be refined through hill-climbing.

A hybrid algorithm requires care in balancing the various components of the search procedure. Indeed, if high-rated solutions are injected in the population at an early evolution stage, this may adversely affect population diversity and force the evolution in wrong directions. The way to avoid this depends, in general, on the particular optimization technique introduced. If a gradient based technique is used as a solution refinement operator, it may be useful to balance the improvement rate of both techniques by stopping the hill climber after a few iterations. However, the iteration number choice is mostly a question of practice and experience, and it should take into account both optimization algorithm and problem features.

Another point that requires special care when using hybrid techniques is encoding. Often, indeed, a specialized optimization technique relies on a particular encoding of the problem variables that may play a key role in the efficiency of the method. Therefore, whenever possible, it is worthwhile to extend to the genetic algorithm

Paper presented at the RTO AVT Course on "Intelligent Systems for Aeronautics", held in Rhode-Saint-Genèse, Belgium, 13-17 May 2002, and published in RTO-EN-022.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
01 JUN 2003	**N/A**	**-**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Airfoil Design Using Navier-Stokes Equations and Hybrid Evolutionary Optimization Techniques	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Centro Italiano Ricerche Aerospaziali Via Maiorise, 81043 Capua, Italy**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, distribution unlimited

13. SUPPLEMENTARY NOTES
See also ADM001519. RTO-EN-022, The original document contains color images.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**UU**	**14**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

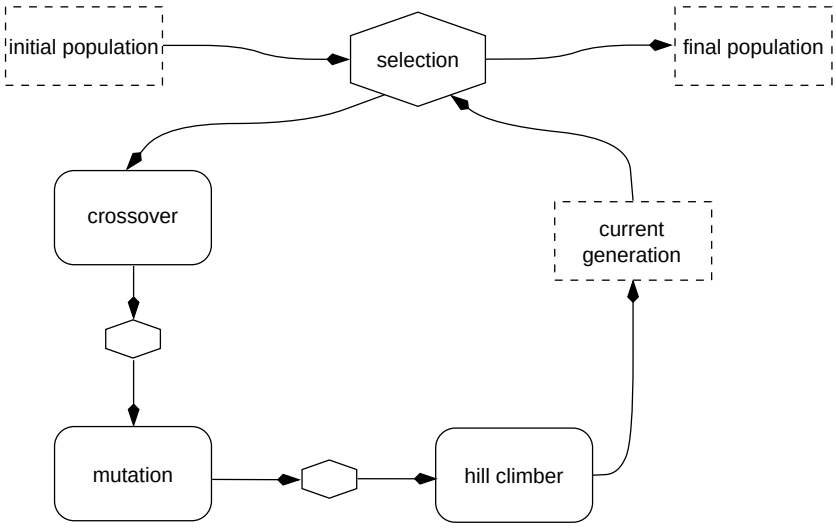

Figure 1: A simple hybridization scheme between a genetic algorithm and a hill climber.

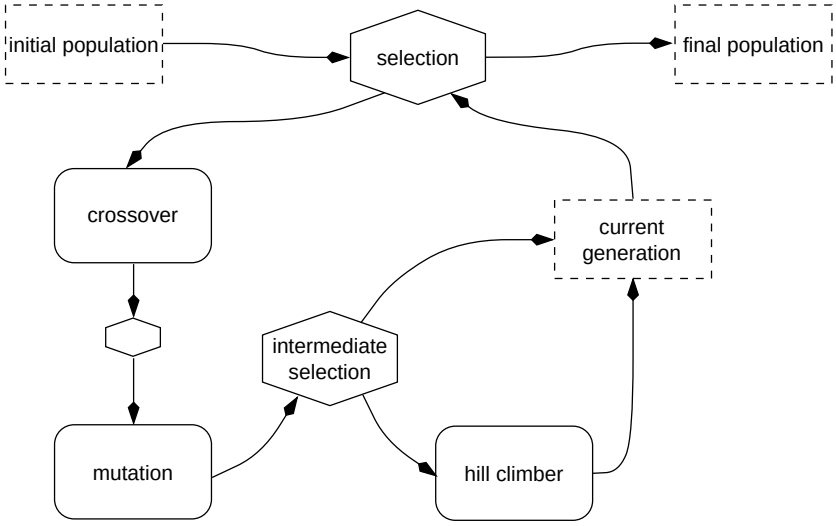

Figure 2: A more complex genetic-hill climber coupling scheme.

the encoding of the specialized technique. On the other hand, the simple binary encoding used by many genetic algorithms may cause problems to the specialized algorithms if the conversion of data is not carefully handled. For example, if a binary genetic algorithm does not use enough bits for variable encoding, the improvement obtained using a gradient based hill climber may be lost when re-encoding the variables in binary form.

The hill climber operator adopted here is a gradient search based on BFGS algorithm [7]. Indeed, the fitness functions of the presented applications are differentiable, for which gradient based techniques are much more efficient to locally improve a given solution. The genetic algorithm developed adopts a bit string codification of the design variables; anyway, this does not prevent the use of operators requiring real number list encoding, such as extended intermediate crossover and word level mutation [8]. In this cases, the binary string is decoded into a real numbers list, the operator is applied and the modified variables set is encoded back into a bit string. This scheme allows the use of a free mix of different type of operators. The hybrid genetic algorithm operates as follows: through the application of the selection, crossover and mutation operators, an intermediate generation is created from the current one; afterwards, if the hybrid option is activated, some randomly chosen individuals are fed into the BFGS based operator to be improved, and then introduced into the new generation.

3 Parallel genetic algorithm

The hybrid genetic algorithm here described has two points that can be executed in parallel: the evaluation of the new population members after the mutation and recombination phase and the evaluation of the gradient through finite differences in the gradient based hybrid operator. The parallel programming model adopted relies on shared memory multiprocessing and the parallelism is implemented at the thread level using the standard POSIX thread interface.

The same code base works on a SGI POWER CHALLENGE system with 16 R-10000 processors and on a Linux PC-cluster with eight processors and the MOSIX clustering software [9].

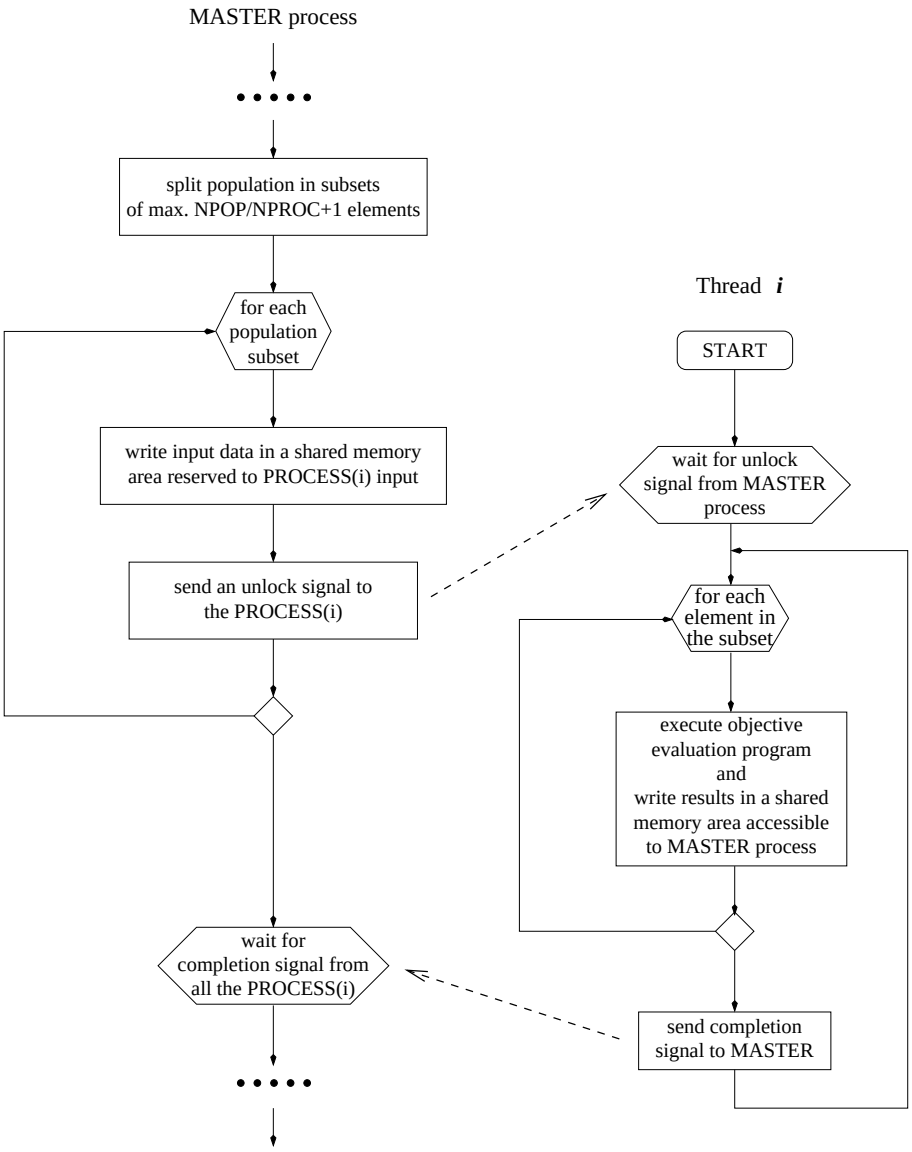

Figure 3: Parallel evaluation loop.

The algorithm is organized following the master-slave paradigm. Figure 3 shows the architecture of the parallel evaluation loop.

In the initialization phase that precedes the first execution of the evaluation loop, the master process creates a pool containing a number of processes, equal to the maximum number of threads available for the computation (NPROC). The child threads are immediately put in a wait state, and they will remain in such a state until they receive a “go ahead” signal from the master to start the computation. This choice avoids the inefficiency of creating a child process every time a computation is needed and killing it at the end.

When the master process enters the evaluation loop, it splits the population, of size NPOP, in slices of maximum NPOP/NPROC+1 elements and then each slice is assigned to one of the child threads. Afterwards, a signal is sent to the child, through a standard POSIX semaphore, so that it begin to evaluate its slice. When the child terminates its computation, it sends a completion signal to the master (using another POSIX semaphore). The master waits for the completion of all child processes in a synchronization point. The child processes are terminated at the very end of the program, when all the evaluation loops related to each generation and gradient evaluation loop have been completed.

This architecture has maximum efficiency when each thread has an even computational charge. If this is not the case, the computation process can loose efficiency because the master has to wait at the synchronization point for the completion of the slowest process. A partial solution to this problem consists in activating more threads than available processors (e.g. 8 threads on a four processor machine). If this is not possible, then the algorithm should be modified in order to redistribute dynamically the computational charge to the child threads.

4 Aerodynamic flow solvers

Two flow solvers with different level of accuracy have been adopted for the objective function computation of the presented design exercises. The first one is the in-home developed ZEN Navier-Stokes solver [10] with Baldwin Lomax turbulence model [11]. The second one is Drela's MSES code [12] that is based on a finite-volume discretization of the Euler equations on a streamlined grid. The viscous region is computed using an integral boundary layer based on a multi-layer velocity profile representation. The inviscid and viscous regions are coupled using displacement thickness.

The boundary layer code is used both as a low-fidelity solver, and as a helper for ZEN code. Indeed, when constant lift coefficient (c_l) is required, a suitable angle of attack for ZEN is guessed using MSES; after a first computation with ZEN, the angle of attack is corrected using the $\partial c_l/\partial\alpha$ computed with MSES.

5 RAE 2822 airfoil optimization

The optimization problem requires the minimization of the drag coefficient $obj = c_d$, with control on lift coefficient and maximum thickness. The design point is fixed at: $M = 0.68$, $Re = 5.7 \times 10^6$, $c_l = 0.56$. The starting geometry is the RAE 2822 airfoil. Transition is fixed to $x/c = 0.01$.

The airfoil shape is defined as a linear combination of an initial geometry $y_0(x)$ and some modification functions, $f_i(x), i = 1, \ldots, n$:

$$y(x) = y_0(x) + \sum_{i=1}^{n} w_i f_i(x) \tag{1}$$

where w_i are the design variables.

The modification functions used here are reported in Table 1 with $\xi = x/c \in [0, 1]$.

5.1 Optimization using simple GA and Euler+BL

A first optimization run has been performed using the genetic algorithm without hybrid operators, and the MSES solver only. The geometry was represented using 20 design variables (10 for the upper surface and 10 for the lower) chosen among the polynomial, Hicks-Henne and Wagner functions. Three subsequent runs of the parallel GA were performed, and 10 threads were active in each run. The GA parameters are reported below:

Variables encoding	32 bit
Selection	3 step R. W.
Crossover	one-point, $p_c = 1$
Mutation	bit level, $p_m = 0.01$
Population size	40
Generations	40

Hicks-Henne	Legendre
$0.888\,(1-\xi)\,\sqrt{\xi}e^{-13.28\xi}$ $0.57\,(1-\xi)\,\sqrt{\xi}e^{-5\xi}$ $0.1\sin^3\left(\pi\xi^{0.431}\right)$ $0.1\sin^3\left(\pi\xi^{0.757}\right)$ $0.1\sin^3\left(\pi\xi^{1.357}\right)$ $0.1\sin^3\left(\pi\xi^{3.106}\right)$	$0.42\,(1-\xi)^3\,\sqrt{\xi}$ $0.946\,(1-\xi)^3\,\xi$ $0.136\,(1-\xi)^3\left(12\xi-10\xi^2\right)$ $(1-\xi)^3\left(225\xi^5-630\xi^4+560\xi^3-220\xi^2+30\xi\right)$
Linear	**Wagner**
0.2ξ	$0.87\left(\frac{2\arcsin\left(\sqrt{\xi}\right)+\sin\left(2\arcsin\left(\sqrt{\xi}\right)\right)}{\pi}-\xi\right)$ $0.24\left(\frac{\sin\left(2k\arcsin\left(\sqrt{\xi}\right)\right)}{k\pi}+\frac{\sin\left(2(k-1)\arcsin\left(\sqrt{\xi}\right)\right)}{\pi}\right)\ k=2,\ldots,6$
Polynomial	**Rear loading**
$0.52\left(0.5\xi^3-1.5\xi^2+\xi\right)$ $0.4\left(\xi-\xi^2\right)$ $0.52\left(0.5\xi-0.5\xi^3\right)$	$6625000\,(1-\xi)\,\xi^{15}e^{1/5-20\xi}$ $17500000\,(1-\xi)\,\xi^{18}e^{1/4-20\xi}$ $44440000\,(1-\xi)\,\xi^{22.66}e^{7/20-20\xi}$ $90000000\,(1-\xi)\,\xi^{30}e^{1/2-20\xi}$

Table 1: Modification functions used in the design examples.

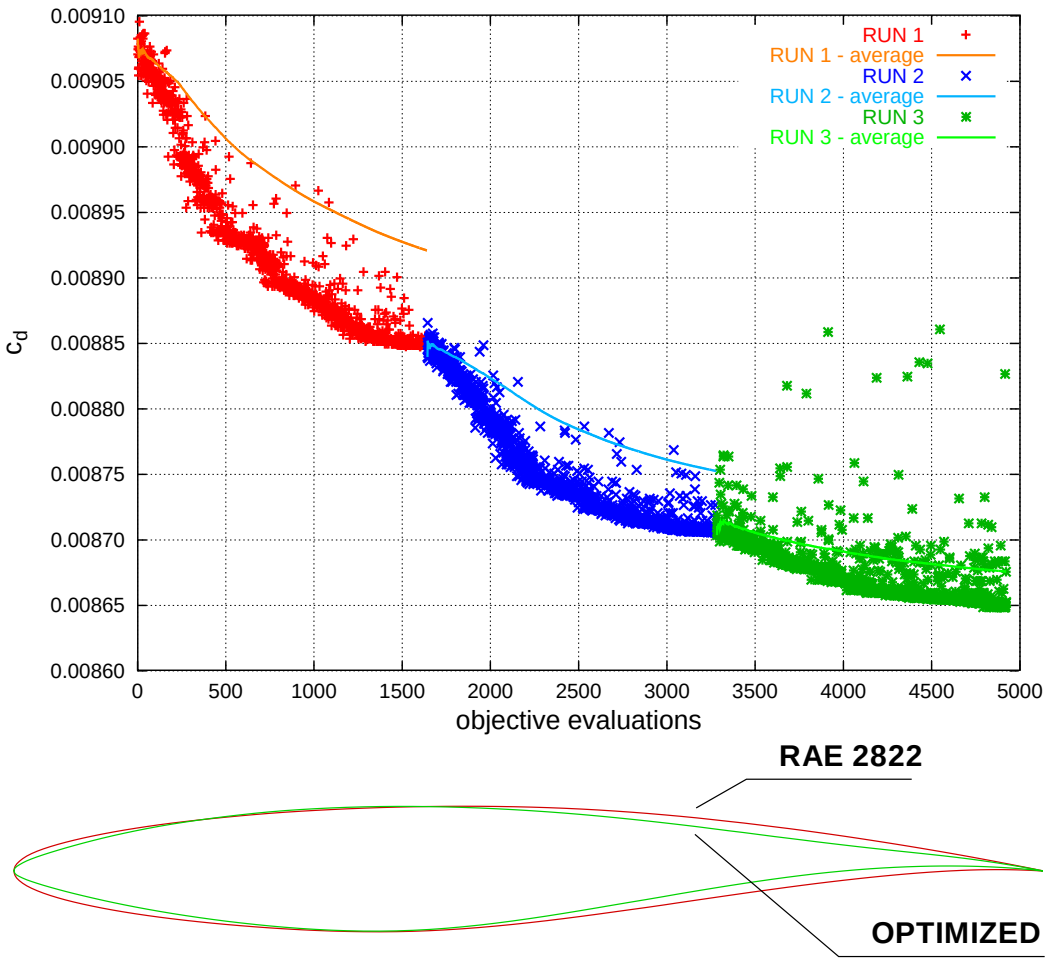

Figure 4: Optimization history and obtained shapes in the Euler+Bl optimization runs.

The optimization history, and the initial and final shapes are reported in figure 4. The initial drag coefficient was $c_d = 0.009070$ at $\alpha = 1.9216°$, while the final one resulted to be $c_d = 0.008645$ at $\alpha = 1.5619°$. A subsequent analysis with the ZEN NS flow solver on the same configurations, however, showed different trends, and there was no appreciable difference between the drag coefficients of the two configurations. Therefore, it was decided to use the initial RAE 2822 as starting point for the subsequent optimization pass with the Navier-Stokes solver.

5.2 Optimization using hybrid GA and NS

The result of the previous run were used to select the design variables that had the strongest effect on the objective function.

A 256×56 C-shaped grid was used for each Navier-Stokes run. A three level multigrid acceleration strategy was used.

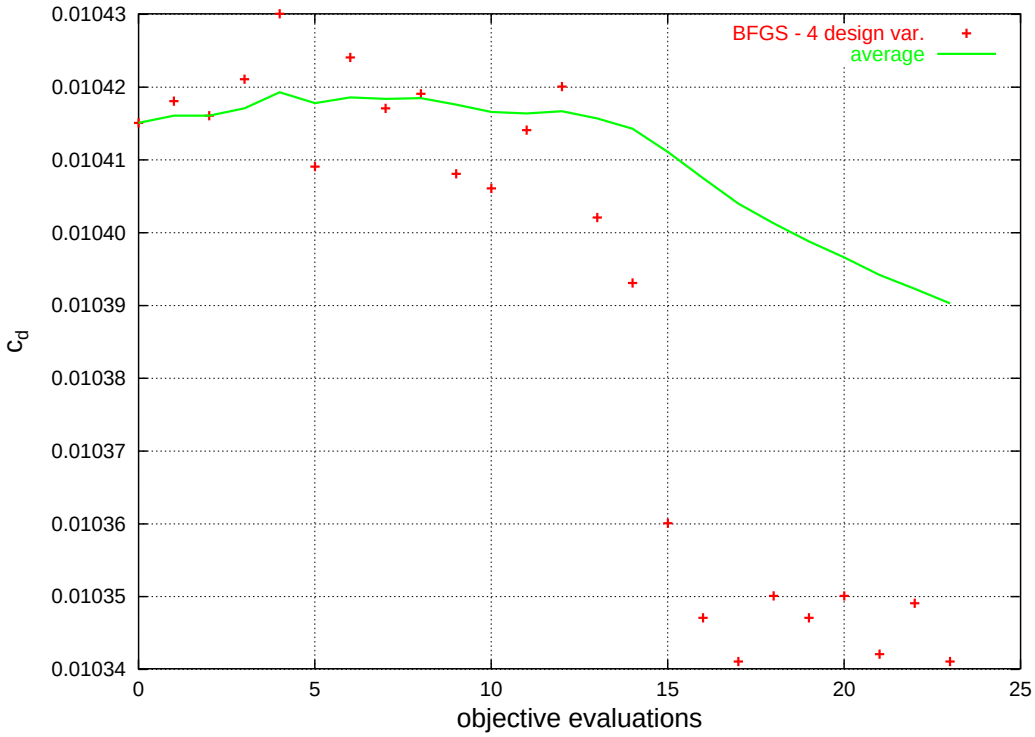

Figure 5: Optimization history with BFGS algorithm, Navier-Stokes equations for drag evaluation and four design variables.

A first set of 4 design variables was identified and a simple BFGS run was performed. The maximum number of BFGS steps was fixed to 10. The optimization history is reported in figure 5. The optimization process terminated when the convergence criteria were meet, after three gradient evaluations and a total of 23 objective function evaluations. As can be observed, a small drag reduction was obtained (0.7%). It is worth to note that in figure 5, as well as in all the following evolution histories reported, all the objective function evaluations required by the optimization algorithm are reported, including those used to compute the gradient by finite differences.

The number of design variables was then extended to eight. Here three different run were performed, The first one was a simple GA with 8 population members that ran for 15 generations. The second one was a hybrid GA that ran for 4 generation, with a population size of two elements. The BFGS was applied to each element, and the number of BFGS steps was fixed to 10. This produced a total of 62 objective function evaluations. The last hybrid GA was instead characterized by a population of eight elements and the BFGS activation probability was set to 6% with three descent steps allowed. After 7 generation the optimizer required 82 objective calculations. The evolution history for these three runs is reported in figure 6. In run 3 the best solution had a drag

coefficient equal to 0.010295 while the starting one was $c_d = 0.010415$. The c_p distribution of the original and the optimized airfoil are reported in figure 7, the c_f comparison is reported in figure 8, and, finally, the Mach field around the airfoil is reported in figure 9.

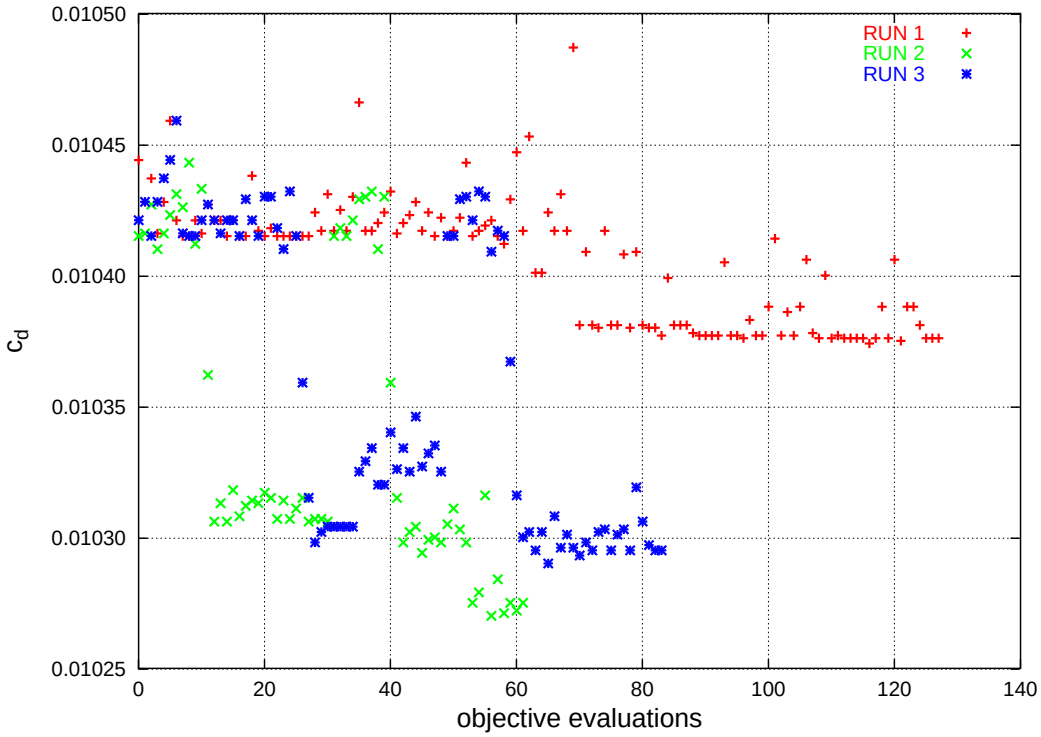

Figure 6: Evolution history of the three runs with eight design variables and Navier-Sokes solver.

6 Use of the adjoint method for gradient computation

A planned improvement of the hybrid genetic algorithm is the introduction of the adjoint method in gradient computation. The major advantage of this approach is that there a higher computational efficiency is obtained. Indeed, by solving the adjoint equations, one obtains all of the gradient components.

A multi-block continuous Navier-Stokes adjoint equation solver for two-dimensional fields was implemented. The numerical solution of the adjoint equations is obtained by using a first-order time-dependent technique based on a finite volume discretization. The solver computes the fluxes at cell interfaces using a flux-vector splitting technique. In a similar way, the boundary conditions are imposed on the numerical fluxes at the computational field edges.

At the moment the adjoint based procedure is being developed as a stand-alone code, whose scheme is reported in figure 10. The gradient computed using the adjoint equation set is then used by a conjugate gradient optimization routine. After the validation phase, the developed solver will be used as core of a new hybrid operator of the genetic algorithm.

The functional to be minimized is:

$$\mathcal{L} = \omega_1 D + \omega_2 \frac{(L - L^*)^2}{2}$$

where D is the drag L is the lift, L^* is the desired lift and the ω_i are weights.

The first case considereds corresponds to the the case of fixed lift without constraints on the geometry. The absence of constraints on shape will lead, as will be evident in the first example reported, to a clear tendency to thickness reduction and to rear loading the airfoil.

Figure 11 reports some results related to the design problem previously defined (RAE2822, $M = 0.68$, $Re = 5.7 \times 10^6$, $c_l = 0.56$) when no control on maximum thickness is imposed. In particular, drag, penalty

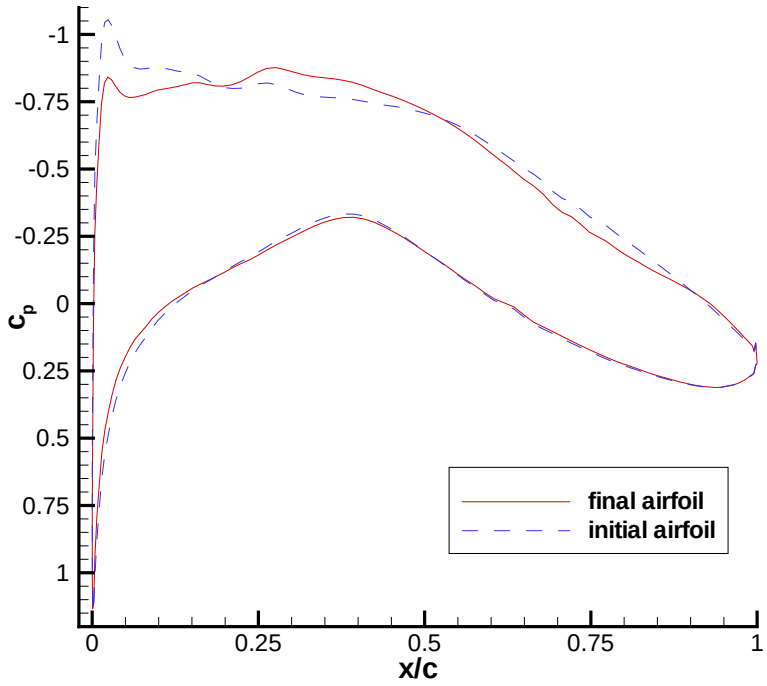

Figure 7: Comparison between initial and final c_p distributions in NS run 3.

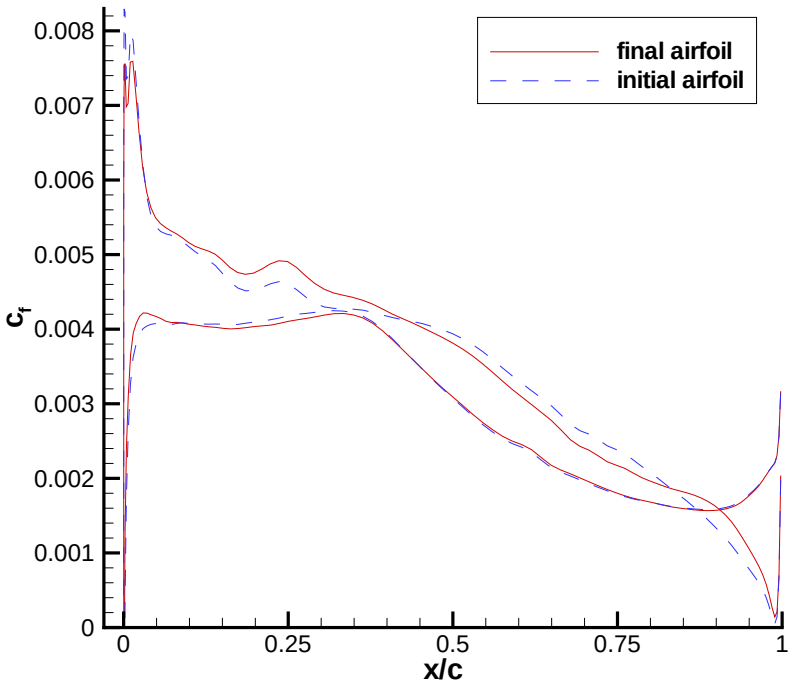

Figure 8: Skin friction coefficient comparison in NS run 3.

on lift, and initial and final airfoil shapes are reported. The gradient is computed point-wise, i.e. each grid point

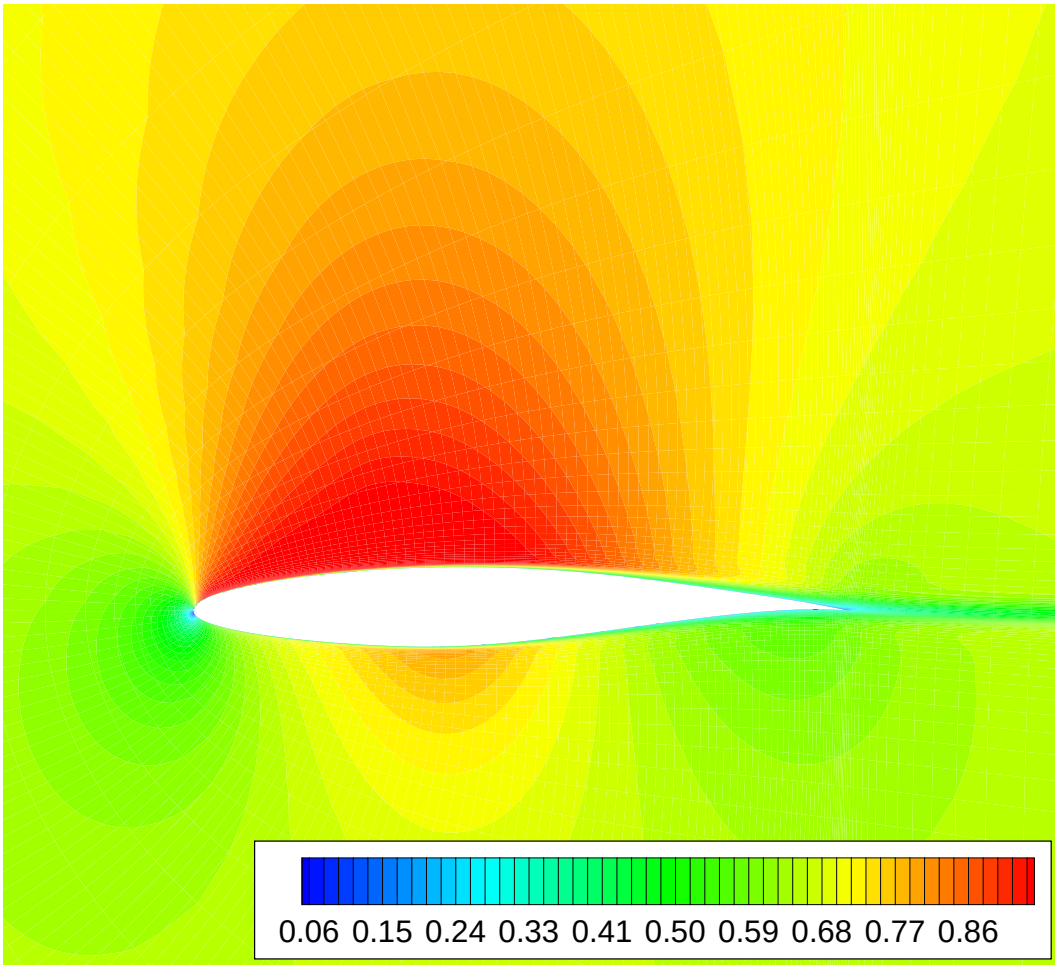

Figure 9: Mach field around the optimized airfoil of run 3.

on the airfoil boundary is a design variable. The gradient analytic expression involves second as well as third derivatives of the flow variables. As the flow field is only second order accurate in space, it can occur that the computation of such derivatives is not reliable in regions of abrupt flow changes, as for example near the front stagnation and close to the rear separation. For this reason the point-wise gradient was smoothed using a Fourier filter. After such filtering the conjugate gradient method is able to nicely decrease the functional $\mathcal{L}$.

Figure 12, instead, reports the result obtained when an a-posteriori control on maximum thickness is imposed. This is obtained projecting, at each iteration, the airfoil modification vector along a direction in which the modification in the maximum thickness section is zero. It is worth to note that while in the first run over 400 field computations were allowed, in this second one only about 160 evaluations were allowed in order to save computational resources.

7 Conclusions

Navier-Stokes flow solvers can be profitably used for aerodynamic shape design with evolutionary optimizers, provided that attention is paid to computational efficiency. In particular, a pre-analysis of the design space with lower fidelity tools is highly recommended to avoid waste of computational resources.

Furthermore, parallel computing and advanced approaches like gradient computation through adjoint, pave the way for challenging industrial applications.

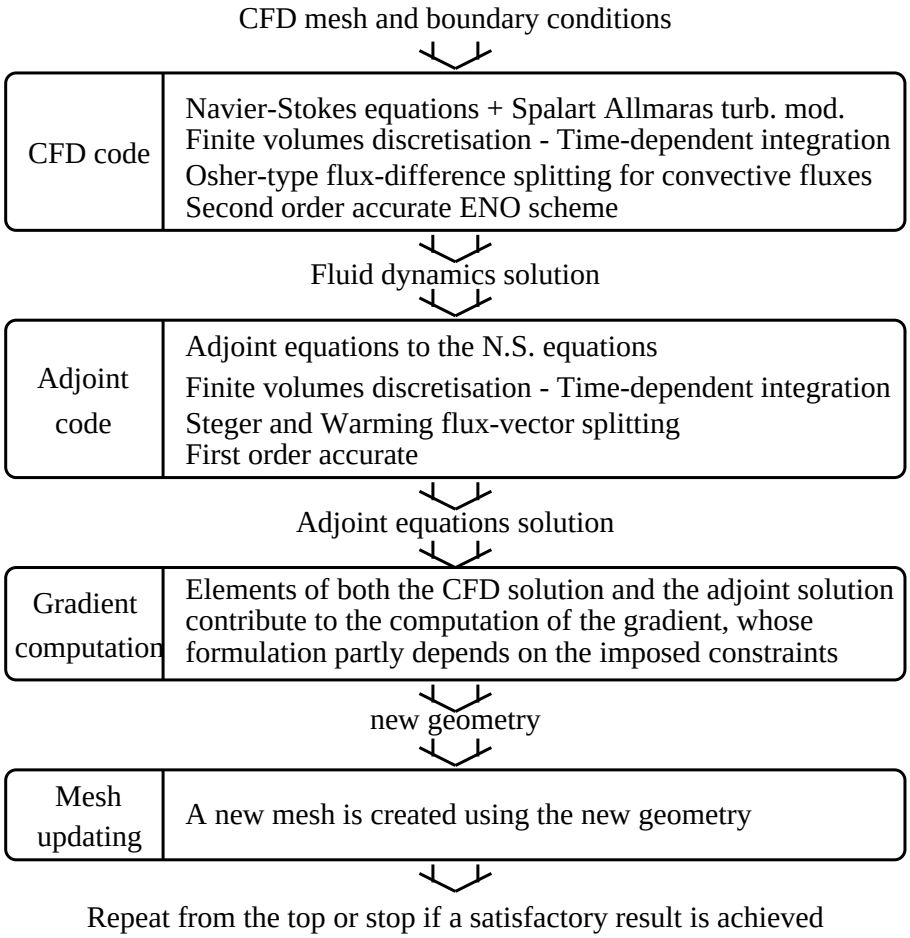

Figure 10: Adjoint optimizer scheme.

Acknowledgments

Thanks are due to prof. Mark Drela for allowing the use of his MSES code in our research work, and to dr. Pietro Catalano and dr. Luigi Paparone of the Italian Center for Aerospace Research (CIRA) for their precious help on the use of ZEN flow solver.

Part of the work here presented was carried out within the framework of the AEROSHAPE project. This project (Multi-Point Aerodynamic Shape Optimization) is a collaboration between Aerospatiale Matra Airbus, Alenia Aeronautica (Coordinator), Daimler Crysler Airbus, EADS-M, Dassault Aviation, SAAB, SENER, SYNAPS, CIRA, DERA, DLR, FFA, INRIA, HCSA, NLR, ONERA, and NuTech Solutions. The project is funded by the European Commission, DG Research, under the GROWTH initiative (Project Ref.: GRD1-1999-10752).

References

[1] Shigheru Obayashi. Pareto genetic algorithm for aerodynamic design using the navier-stokes equations. In Domenico Quagliarella, Jacques Périaux, Carlo Poloni, and Gabriel Winter, editors, *Genetic Algorithms and Evolution Strategies in Engineering and Computer Science*, pages 245–266, England, November 1997. John Wiley & Sons Ltd.

[2] N. Marco, S. Lanteri, J. A. Desideri, and Jacques Periaux. A parallel genetic algorithm for multi-objective optimization in computational fluid dynamics. In Kaisa Miettinen, Marko M. Makela, Pekka Neittaanmaki, and Jacques Periaux, editors, *Evolutionary Algorithms in Engineering and Computer Science*, pages 445–455, England, 1999. John Wiley & Sons Ltd.

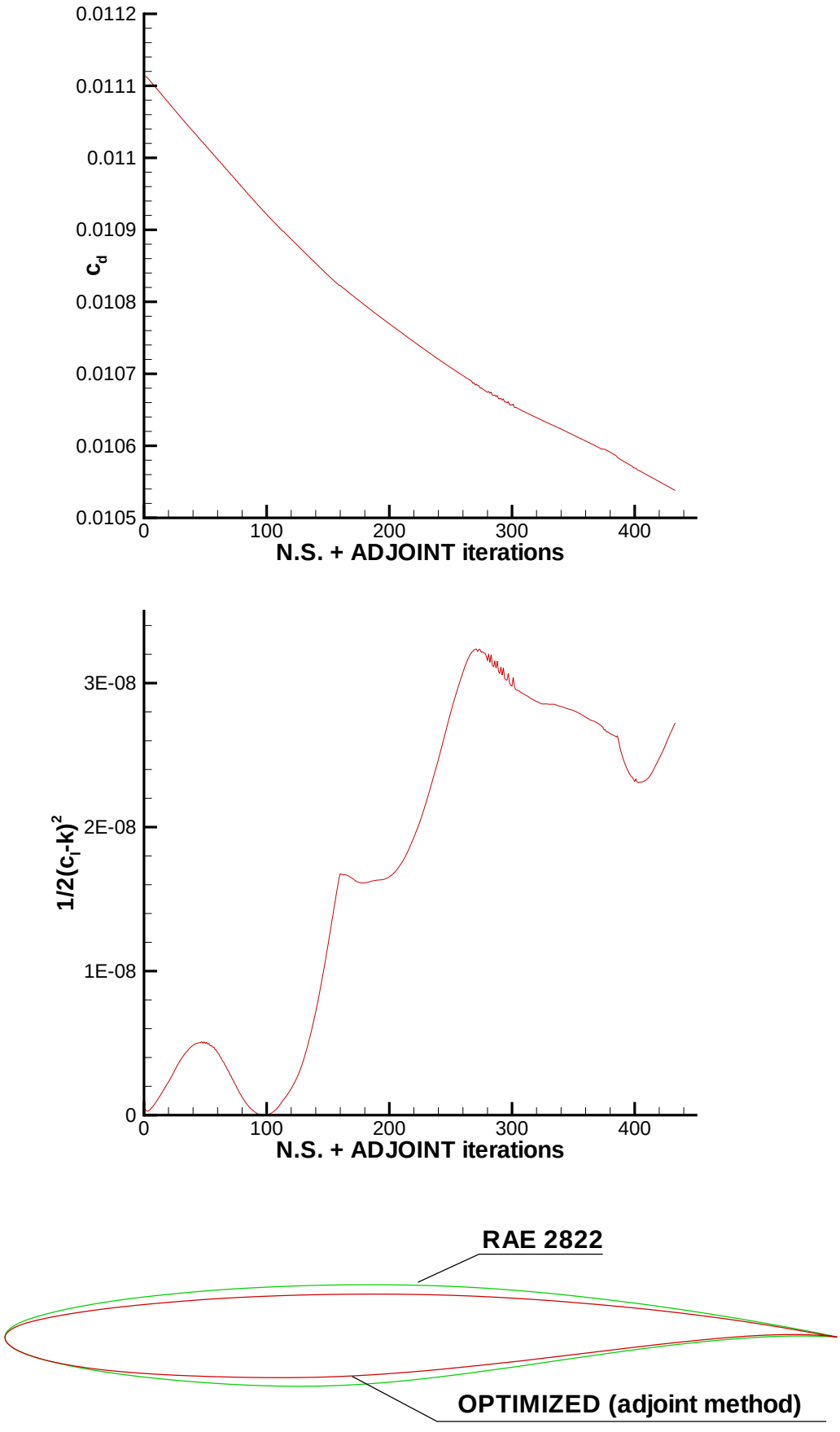

Figure 11: Drag, lift penalty constraint for the functional $\mathcal{L}$, and airfoil shapes comparison for the case without constraint on maximum thickness.

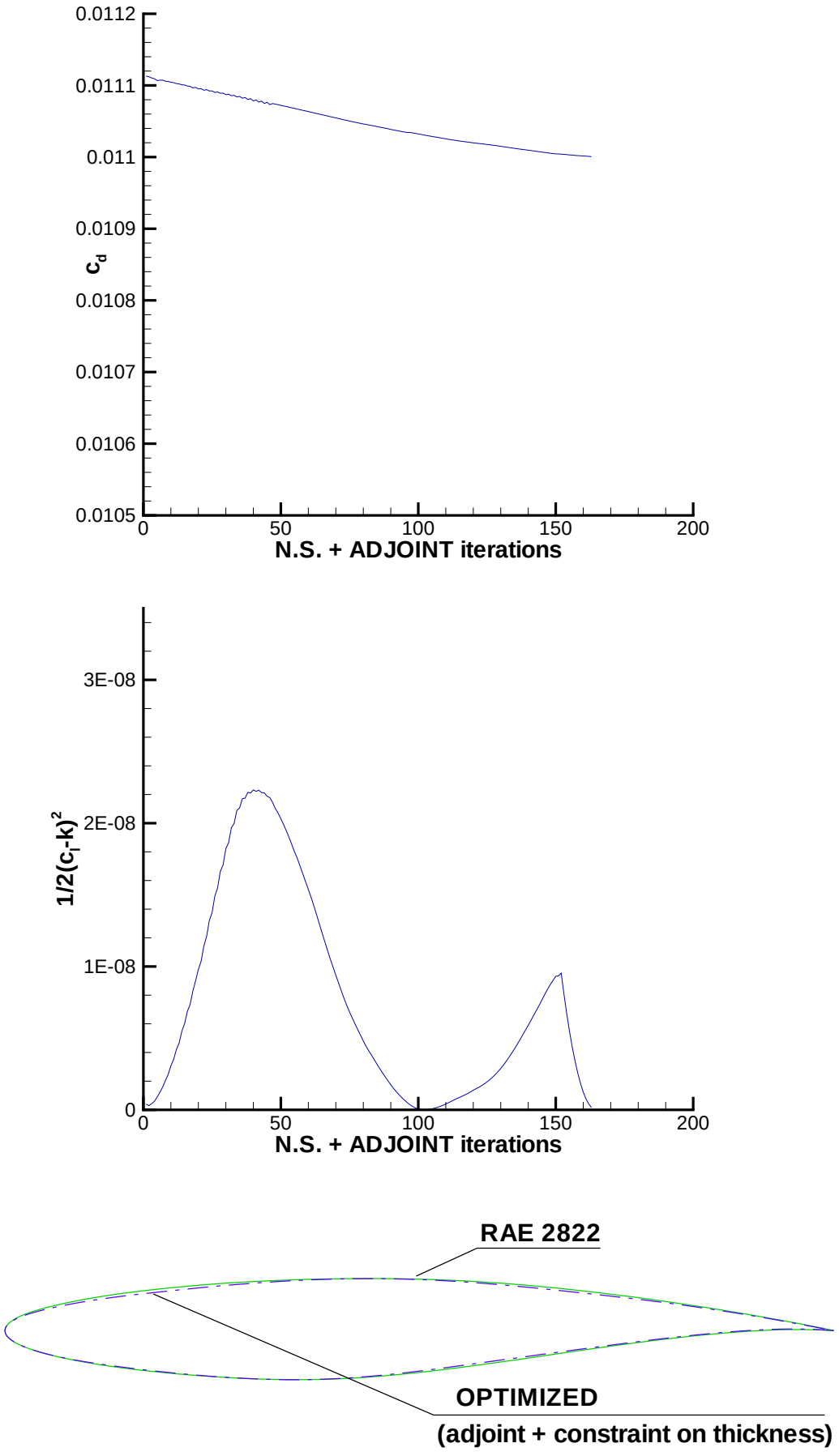

Figure 12: Drag, lift penalty constraint for the functional $\mathcal{L}$, and airfoil shapes comparison for the case with constrained maximum thickness.

[3] A. P. Giotis, K. C. Giannakoglou, and Jacques Périaux. A reduced-cost multi-objective optimization method based on the pareto front technique, neural networks and pvm. In *Proceedings of ECCOMAS 2000 Conference*, Barcelona, Spain, September 11–14 2000.

[4] Carlo Poloni and Valentino Pediroda. Ga coupled with computationally expensive simulation: Tools to improve efficiency. In Domenico Quagliarella, Jacques Périaux, Carlo Poloni, and Gabriel Winter, editors, *Genetic Algorithms and Evolution Strategies in Engineering and Computer Science*, pages 289–309, England, November 1997. John Wiley & Sons Ltd.

[5] Domenico Quagliarella and Alessandro Vicini. Coupling genetic algorithms and gradient based optimization techniques. In Domenico Quagliarella, Jacques Périaux, Carlo Poloni, and Gabriel Winter, editors, *Genetic Algorithms and Evolution Strategies in Engineering and Computer Science*, pages 289–309, England, November 1997. John Wiley & Sons Ltd.

[6] L. Davis. *Handbook of genetic Algorithms*. Van Nostrand Reinhold, New York, 1991.

[7] G. N. Vanderplaats. *Numerical Optimization Techniques for Engineering Design: with Applications*. Mc Graw–Hill, 1984.

[8] Alessandro Vicini and Domenico Quagliarella. Inverse and direct airfoil design using a multiobjective genetic algorithm. *AIAA Journal*, 35(9):1499–1505, September 1997.

[9] A. Barak and O. La'adan. The mosix multicomputer operating system for high performance cluster computing. *Journal of Future Generation Computer Systems*, 13:361–372, March 1998.

[10] J. C. Kok, M. Amato, and I. W. Boerstoel. Mathematical-Physical modeling for multi-block NaS/Euler simulation. Technical report, Centro Italiano Ricerche Aerospaziali, Capua, Italy. CIRA-DLCEST-TR183, and NLR-CR91-235L.

[11] B. S. Baldwin and H. Lomax. Thin layer approximation and algebraic model for separated turbulent flows. American Institute of Aeronautics and Astronautics (AIAA), 1978. AIAA Paper 78-257.

[12] Mark Drela. Newton solution of coupled viscous/inviscid multielement airfoil flows. In *AIAA, Fluid Dynamics, Plasma Dynamics and Lasers Conference*, Seattle, WA, June 1990. American Institute of Aeronautics and Astronautics (AIAA). AIAA Paper 90-1470.

This page has been deliberately left blank

Page intentionnellement blanche